VŠB – Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science
Department of Applied Mathematics

Image feature extraction applied in classification techniques

Extrakce významných rysů v obrazových scénách s využitím v klasifikačních technikách

2019 Vojtěch Dorňák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Zadání bakalářské práce

Student: **Vojtěch Dorňák**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 1103R031 Výpočetní matematika

Téma: Extrakce významných rysů v obrazových scénách s využitím v klasifikačních technikách
Image feature extraction applied in classification techniques

Jazyk vypracování: čeština

Zásady pro vypracování:

Moderní techniky pro klasifikaci obrazových dat jsou schopné efektivně zpracovávat originální obrazové vstupy tzn. raw image data. Mezi jednu z nejznámějších technik patří umělé neuronové sítě např. TenzorFlow. Hlavní nevýhodou této techniky je neunifikovanost v návrhu struktury neuronové sítě. Proto se i nadále v některých průmyslových aplikacích používají konvenční přístupy strojového učení např. Support Vector Machines (SVM). V případě použití SVM pro klasifikaci netriviálních obrazových scén je však vhodné transformovat obraz do prostoru rysů, ve kterém jsou podchycené významné charakteristiky obrazové scény.
Cílem bakalářské práce je prozkoumat vhodné transformace obrazových dat do prostoru rysů a jejich použitelnost v klasifikační technice SVM. Mohou být vyzkoušené například waveletová transformace či SVD rozklad v redukovaném tvaru. Navržené přístupy budou otestované na průmyslových obrazových datech.

Seznam doporučené odborné literatury:

Častová, N., Horák, D., Kaláb, Z. Description of seismic events using wavelet transfrom, (2006) International Journal of Wavelets, Multiresolution and Information Processing, 4 (3), pp. 405-414.

Gene H. Golub, Charles F. Van Loan, Matrix Computations, JHU Press, 1996.

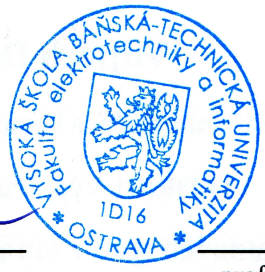
Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Ing. Martin Čermák, Ph.D.**

Konzultant bakalářské práce: Ing. Marek Pecha

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019



prof. RNDr. Jiří Bouchala, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

I hereby declare that this bachelor's thesis was written by myself. I have referenced all the sources I have drawn upon.

Ostrava, 30th April, 2019 .....

I would first like to thank the thesis supervisor doc. Ing. Martin Čermák, Ph.D. for his assistance and patience throughout the process of writing this thesis. I would also like to thank the thesis consultant Ing. Marek Pecha. Throughout every step of writing this thesis, his expertise in the topic of Support Vector Machines and his eagerness to help me explore the topic of Scale Invariant Feature Transform, even at the expense of his sleep, were invaluable. Last but not least, I would like to thank my parents and my girlfriend for their everyday support and patience.

Abstrakt

V rámci této bakalářské práci prozkoumáme metodu extrakce obrazových rysů zvanou Scale-Invariant Feature Transform. Extrahované rysy využijeme k trénování a testování klasifikační techniky zvané Support Vector Machine. V prvních několika sekcích představíme Scale-Invariant Feature Transform, techinky potřebné k přizpůsobení vektorů rysů klasifikátoru a Support Vector Machine. V benchmarcích natrénujeme a otestujeme klasifikátor na rysech extrahovaných z reálných fotografií. Pro porovnání natrénujeme a otestujeme klasifikátor také pomocí hodnot jednotlivých pixelů ve fotografiích.

Klíčová slova: extrakce rysů, Scale-Invariant Feature Transform, klasifikace dat, Support Vector Machine, Bag of Words, Bag of Visual Words, k-means, k-means++

Abstract

In this thesis, we explore image feature extraction approach called Scale-Invariant Feature Transform. We use the extracted features for training and testing of an image classification technique, a Support Vector Machine. In first few sections, we introduce the Scale-invariant Feature Transform, techniques used to adapt the the feature descriptors to the classifier, and the Support Vector Machine. For the benchmarks, we train and test the classifier using features extracted from a real image data. For comparison, we train and test the classifier using pixel values of the data.

Key Words: feature extraction, Scale-Invariant Feature Transform, data classification, Support Vector Machine, Bag of Words, Bag of Visual Words, k-means, k-means++

Contents

List of symbols and abbreviations

SVM	–	Support Vector Machine
SIFT	–	Scale Invariant Feature Transform
LoG	–	Laplacian of Gaussian
DoG	–	Difference of Gaussian
BoW	–	Bag of Words
BoVW	–	Bag of Visual Words
CV	–	Cross-Validation
GS	–	Grid Search
QP	–	Quadratic Programming
FE	–	Feature Extraction
MPRGP	–	Modified Proportioning and Reduced Gradient Projection
MPI	–	Message Passing Interface
HyperOpt	–	Hyperparameter Optimisation

List of Figures

List of Tables

1 Introduction

Image classification is an important task in the computer vision field. It is used in systems for automatic inspection in manufacturing applications, computer-human interaction, information organisation, etc. [1]. The most frequently used classification techniques include artificial neural networks such as TenzorFlow [2]. The biggest disadvantage of this technique is its computational difficulty and the non-unifiability in the neural network structure design. This is why the conventional classification techniques, such as an Support Vector Machine, are still used today.

To classify images, the images can be represented using their raw pixel values. However, it might be desirable to transform the images into a feature space exploiting a feature extraction technique. This thesis aims to explore such technique and to compare the results of classification using the extraction technique to the results using the the raw image data.

The feature extraction technique we utilize is the Scale Invariant Feature Transform. This is described in Section 2. To adapt the features to a classification technique, we use a k-means clustering and Bag of Visual Words approach described in Section 3. For the classification, Support Vector Machine described in Section 4, is used. The classification is then benchmarked and the quality of the classification is discussed in Section 7.

2 SIFT Extraction

In this section, we introduce the Scale Invariant Feature Transform (SIFT) algorithm proposed by David Lowe, and published in the original paper [3] in 2004. Compared to ordinary techniques such as the Harris Corner Detector [4] or Canny edge detector [5], the SIFT identifies the general (not only edges and corners) points of interest, i.e. keypoints, and their feature vectors, which describe objects in an image scene in a more accurate manner.

The extracted feature vectors are scale invariant. Moreover, they are invariant to rotation, illumination and to change in affine transformations. In our text, we introduce the methodology of training the classifier that recognizes the objects in real images (photographs); therefore the properties of SIFT feature vector are essential.

2.1 Keypoint Extraction

First of all, we want to find the candidates for the stable keypoints, i.e. keypoints that can be reproduced across a mixture of image variations. The candidates are determined as local optima of Laplacian of Gaussian (LoG). Then from these candidates, the points which are located in areas with low contrast or close to the edges, are being rejected.

2.1.1 Laplacian of Gaussian

LoG is an operator with a strong response to blobs (arbitrary regions in an image with different brightness compared to its surroundings). This operator is constructed in the following steps.

Firstly, an input image, $I_{img}(x, y)$, is convoluted by the Gaussian kernel

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{1}$$

at a scale σ to get its scale-space representation, so that:

$$L(x, y, \sigma) = G(x, y, \sigma) * I_{img}(x, y). \tag{2}$$

Then, the scale-normalized Laplacian operator is applied:

$$\triangle_{norm} L(x, y, \sigma) = \sigma \left(\frac{\partial^2 L(x, y, \sigma)}{\partial x^2} + \frac{\partial^2 L(x, y, \sigma)}{\partial y^2} \right). \tag{3}$$

Determination of the local extrema of (3) would be time consuming, therefore, it is proposed to approximate by the Difference of Gaussian (DoG) [3].

The consequence that DoG approximates LoG can be shown by heat diffusion equation, i.e:

$$\frac{\partial L}{\partial t} = k \triangle L. \tag{4}$$

We assume that k (thermal diffusivity) is changing in time t, so that

$$\frac{\partial L}{\partial t} = k(t) \triangle L \tag{5}$$

where

$$k(t) = t, t \geq t_0 > 0. \tag{6}$$

Using σ instead of time t, we obtain

$$\frac{\partial L}{\partial \sigma} = \sigma \triangle L. \tag{7}$$

Then we approximate $\frac{\partial L}{\partial \sigma}$ by means of forward differences with a factor k, so that

$$\frac{\partial L}{\partial \sigma} \approx \frac{L(x, y, k\sigma) - L(x, y, \sigma)}{k\sigma - \sigma}. \tag{8}$$

Resulting from (7) and (8), we obtain the required approximation:

$$(k-1)\sigma^2 \triangle L \approx L(x, y, k\sigma) - L(x, y, \sigma) =: D(x, y, \sigma), \tag{9}$$

where $D(x, y, \sigma)$ denotes DoG.

From equation (9), we can see that DoG already incorporates σ^2 scale-normalization, using the constant factor k. In Figure 1, we can see the comparison of the DoG and LoG (Mexican hat wavelet in case of multi-dimension generalisation).

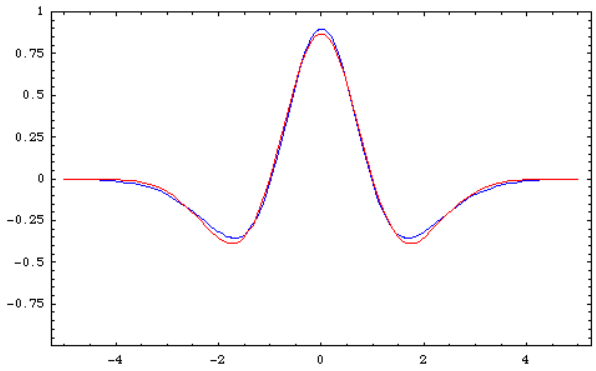


Figure 1: A comparison of a Mexican Hat Wavelet and DoG. [6]

2.1.2 Octaves

In order to locate stable keypoints, we find the local optima not only in the domain of one DoG, however it is found across different scales as well. The efficient approach to creating DoGs at the scales is progressively convolving the original image with the Gaussian kernel, where consecutive Gaussian kernels differ in a multiplicative constant factor k. The optimal choice is $k = \sqrt{2}$ (suggested by the "father" of the SIFT algorithm [3]).

We generate images at $s+3$ scales, where s is chosen such that $k = 2^{1/s}$; in our case $s = 2$. It is called the octave. For $k = \sqrt{2}$, we obtain 5 scales in the octave. In the next octave, the resolution of an image (at scale double the original value of σ) is reduced to half of the original resolution, so that every second row and column are removed. Using the resized image, new octave is generated. Commonly, 3 octaves are formed. In each octave, we subtract images in consecutive scales in order to construct DoG. This can be seen in Figure 2.

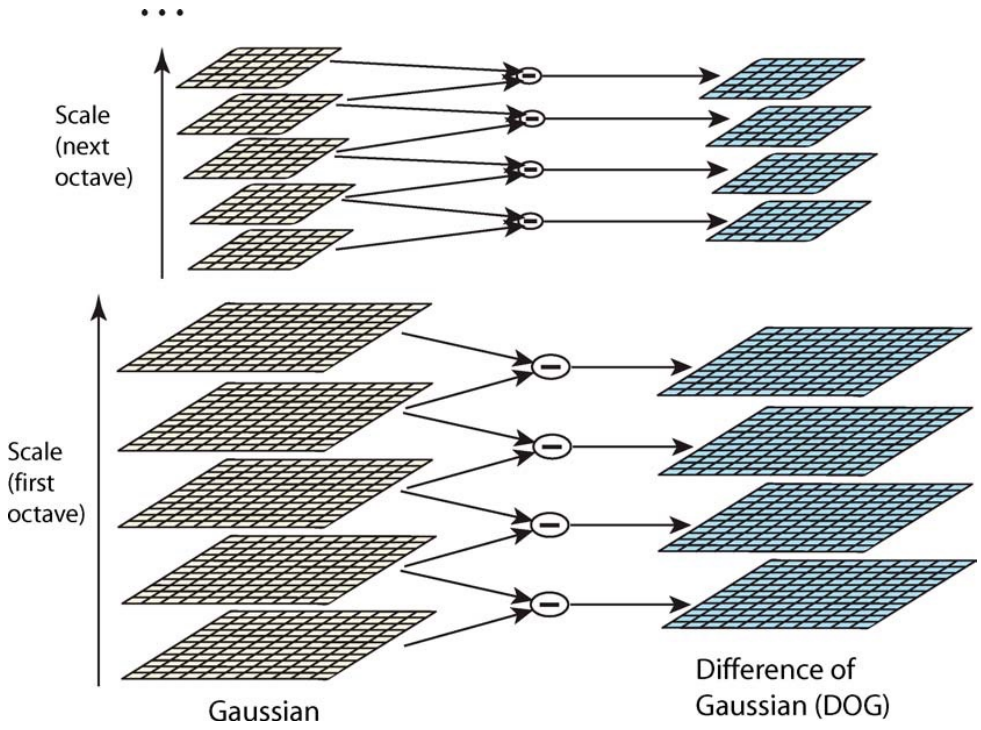


Figure 2: DoG pyramid. [3]

To determine the keypoint candidates, we compare each pixel value with the values of 8 pixels surrounding it in the current scale, and with pixel values in the 3×3 areas in scales right below and right above, respectively. This can be seen in Figure 3. The pixel is selected as a keypoint candidate if it has higher or lower value than these 26 pixels $(8 + 9 + 9)$.

To detect a keypoint candidate in its subpixel position, we interpolate nearby data and locate the interpolated extremum. For interpolation we exploit the quadratic Taylor expansion of the DoG with the keypoint candidate as its origin:

$$D(\boldsymbol{x}) = D + \frac{\partial D^T}{\partial \boldsymbol{x}}\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\frac{\partial^2 D}{\partial \boldsymbol{x}^2}\boldsymbol{x}, \tag{10}$$

where $x \in \mathbb{R}$, $D(x, y, sigma)$ and the partial derivatives are evaluated at this keypoint, $\boldsymbol{x}$ is the offset from the keypoint. Then, extremum $\hat{\boldsymbol{x}}$ is located by setting the gradient of D to be zero vector:

$$\hat{\boldsymbol{x}} = -\frac{\partial^2 D^{-1}}{\partial \boldsymbol{x}^2}\frac{\partial D}{\partial \boldsymbol{x}}. \tag{11}$$

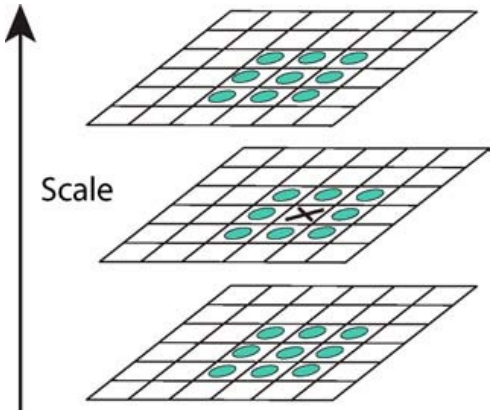


Figure 3: Optima of the DoG are selected by the means of comparing pixel to the 26 pixels surrounding it in the scale-space. [3]

If the offset $\hat{\boldsymbol{x}}$ is larger than 0.5 in any dimension, it indicates that the extremum is close to another candidate keypoint and is discarded. In the case the offset is not larger than 0.5 in any dimension, it is added to the keypoint candidate to get an extremum in subpixel in the . Then we compute the extremum in the new subpixel position and then we determine the extremum in the new subpixel position

2.1.3 Choosing Suitable Keypoints

If the area around the keypoint candidate has a low contrast, the keypoint is sensitive to noise, or it is poorly localized along an edge. Then the keypoint is rejected.

Low contrast keypoint is detected by calculating (10) at the location of the keypoint $\hat{\boldsymbol{x}}$. Assuming pixel values that are normalised between 0 and 1, we discard the keypoint if the value of $D(\hat{\boldsymbol{x}})$ is evaluated as lower than 0.03 (0.04 is used in OpenCV). This value has been experimentally determined in [3].

In order to detect poorly localized keypoints along an edge, we need to take look at their principal curvatures. The poorly localized keypoints have principal curvature perpendicular to an edge much larger than the principal curvature along it. To find the principal curvatures, we compute the eigenvalues of Hessian matrix H:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}, \tag{12}$$

where the partial derivatives are approximated by means of differences of neighbouring keypoints with the current keypoint.

For the SIFT purposes, just the ratio of the eigenvalues is required. Let us denote α as the larger eigenvalue and β as the smaller one. Sum of these eigenvalues is defined as the trace of H, i.e.

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta, \tag{13}$$

and the product of the eigenvalues as the determinant of H, i.e.

$$Det(H) = D_{xx}D_{yy} - D_{xy}^2 = \alpha\beta. \tag{14}$$

Let us denote the ratio of the eigenvalues as r, such that

$$r = \frac{\alpha}{\beta}. \tag{15}$$

Then,

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}. \tag{16}$$

To inspect that the ratio of eigenvalues is below the value of r, we exploit the following inequality:

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}. \tag{17}$$

In [3] $r = 10$ is proposed.

After removing the unstable keypoint candidates, we are left with keypoints should likely be reproduced across different image conditions.

2.2 Feature Vector Calculation

From previous workflow, we've obtained suitable keypoint locations. We need to find a reliable way to represent the keypoint locations as a descriptor. We need descriptors that are almost identical across different scale, rotation, illumination and by applying the affine transformations.

2.2.1 Keypoint orientation

In order to describe the keypoints invariant to the object rotation correctly, we have to determine its orientation: in our further calculations it ensures rotation invariance for the descriptor. Using the scale of the keypoint, we select Laplacian with Gaussian kernel L with the closest scale. It leads to computations that are scale invariant.

First, for each point within a region around the keypoint, we evaluate its gradient magnitude, so that:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}, \tag{18}$$

and its orientation

$$\theta(x,y) = tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right). \tag{19}$$

Then, orientation histogram with 36 bins (10° per bin) is created, to which sample points are added and weighted by their magnitude and the Gaussian-weighted circular window with σ equal to 1.5 times the scale of the keypoint.

Further, we select largest bin in the histogram as the keypoint orientation. All the other bins that are more than 80% of the largest bin become new keypoints at the same location as the current keypoint in the determined orientation. Now we have all, that we need, for determining the keypoint descriptor.

2.2.2 Keypoint descriptor

Using the keypoint scale σ, Gaussian blur is applied to the image. For each keypoint, 16×16 window surrounding it is used. In the window, the gradient magnitude (18) and the orientation (19) is computed for each point. Then, keypoint orientation is then subtracted from the orientations of these points.

Then the window is divided into 16 (4×4) sub-windows. For each of the sub-windows, we create the 8 bin orientation histogram by means of the keypoints orientations weighted by their magnitude and Gaussian-weighted circular window with σ equal to 1.5 times scale current DoG, see Figure 4.

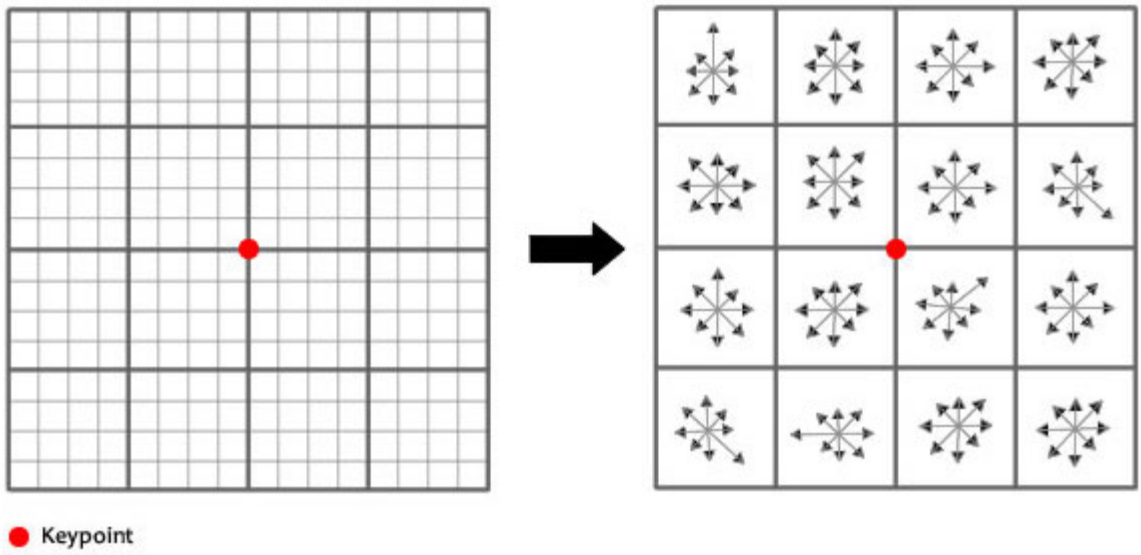


Figure 4: Extracting the sift-descriptor. [7]

These 16 histograms are put to a descriptor vector. For improving invariance to illumination, firstly the descriptor is normalized to unit length, then the threshold of 0.2 is applied, i.e. any value greater than 0.2 is changed to 0.2, and then the vector is normalized again.

From previous workflow, we have obtained the stable keypoints and their descriptor, a vector with dimension 128. We should be able to mach these descriptors across differently oriented images depicting the similar objects that could be taken in non-identical light conditions.

3 Bag of Visual Words

As many other feature extraction (FE) techniques, SIFT generates varied amounts of feature vectors. The locations order of the keypoints, which are described using feature vectors is changed across the images taken from different point of view.

In order to use these feature vectors in the classification techniques, we need their representation as a single vector for each image. Bag of Words (BoW) is one of the most common approaches for obtaining this representation. In case of describing the image scenes, this concept is called the Bag of Visual Words (BoVW). For generating BoVW, we create bins for each one feature vector. From feature vectors of each one image we generate a histogram using these bins.

As feature vectors of the matching keypoints are only similar, we need to group the matching ones together. We create separate BoVW bins for each group, not for each distinct feature vector. These groups can be formed by employing the k-means clustering algorithm.

3.1 k-Means Clustering

The k-means algorithm belongs to the unsupervised learning techniques. Its objective is to divide data points into k clusters based on their similarity. Let $X = \{x_1, x_2, ..., x_n\}$ be a given set of n data points. Formally, the k-means algorithm sorts the n data points out into the k clusters $\boldsymbol{S} = \{S_1, S_2, ..., S_k\}$:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - c_i\|_2^2, \tag{20}$$

where $c_1, c_2, ..., c_k$ are the centroids of $S_1, S_2, ..., S_k$, respectively. The centroids $c_1, c_2, ..., c_k$ are determined as the means of data points belonging to the clusters $S_1, S_2, ..., S_k$.

Firstly, the algorithm starts with selecting the initial centroids. Commonly, the centroids are assigned randomly using either the Forgy or Random Partition methods. The Forgy method selects k random data points as the initial centroids. The Random Partition method assigns each data point to random cluster, then determines the centroids as the means of data points belonging to each cluster.

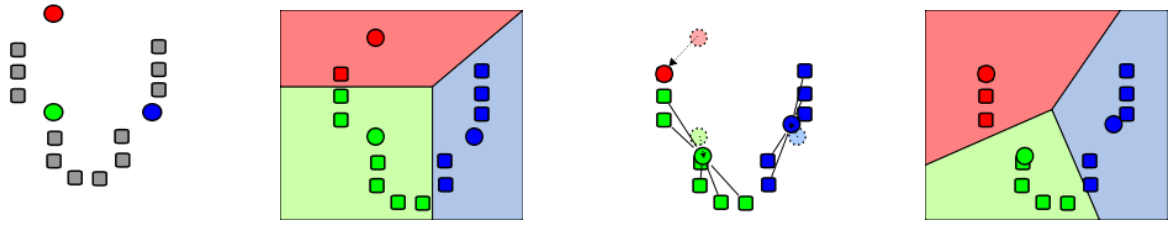
After the initial centroids are selected, the standard k-means algorithm, it is Lloyd's algorithm (see Figure 5), alternates between two steps until the terminate condition is reached:

Assignment step: Each data point is assigned to the cluster, such that the Euclidean distance between the centroids of the clusters and the current data point is the smallest one.

Update step: For each cluster, new centroid is computed as the mean of data points belonging to this cluster.

Usually, the terminate condition is met when the ratio between the samples, which are moved to another cluster, to the total amount of samples is smaller than the threshold. The threshold

is typically set to 1e−5. In OpenCV, the condition, that Euclidean distance between the new centroids and the old ones is smaller than a threshold.



(a) Initial centroids are selected.

(b) By choosing the nearest centroid, each data point is assigned its cluster.

(c) New centroids are calculated as means of data points in the cluster.

(d) Steps Figure 5b and Figure 5c are repeated.

Figure 5: Lloyd algorithm demonstration. [8]

3.1.1 k-means++

In the original algorithm, the random choice of the initial centroids can sometimes lead to poor clustering. Therefore, the k-means++ algorithm has been proposed [9]. It is used for selecting more optimal initial centroids as follows:

1. First centroid is selected from the data points uniformly at random.
2. Using a probability distribution weighted by each data point distances to their nearest centroid, another centroid is selected.
3. The steps 1 and 2 are repeated until k centroids have been chosen.
4. The algorithm continues with a standard k-means algorithm.

Typically, the proposed selection of the initial guess is more time-consuming than the random methods mentioned above. However, the rest of the algorithm usually converges faster using these centroids. This leads to overall lower computation time.

3.1.2 Determining the Number of Clusters

The biggest challenge of using the k-means algorithm is to choose the number of clusters. Standard approach is based on trying it experimentally. In an effort to compare our results with various clusters counts, we analyse the following two metrics:

Calinski-Harabasz Index [10] is seen as a ratio of variance withing clusters to variance between clusters.

Davies-Bouldin Index [11] is computed as a ratio of distances withing clusters to distances between clusters.

The lowest value related to the number of clusters indicates that the number is optimal.

3.2 Bag of Words example

In order to better understand a creation of the BoW, we take a look at following simple example. In this example, we do not generate the BoW using the SIFT features - we use words. Let us consider the following sentences:

1. "John had a salad while James had some buffalo wings."
2. "James, while John had had "had", had had "had had"; "had had" had had a better effect on the teacher." [12]
3. "Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo." [12]

From these sentences, we select each unique word, leaving us with following bins: {"john", "had", "a", "salad", "while", "james", "some", "buffalo", "wings", "better", "effect", "on", "the", "teacher"}. Then we generate a histogram for each of the sentences using these bins:

1. {1, 2, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0}
2. {1, 11, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1}
3. {0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0}

From these histograms, we can figure out how many times each unique word is in a sentence. For example, the word "buffalo" (eight position) appears in the first sentence once, does not appear in the second sentence at all and appears eight times in the third sentence.

As we can see, the sentences are represented by the amounts of each different word in them. They will loose an important information, an order of these words. This does not matter in our case, as the SIFT feature descriptors should not be considered in any particular order.

3.3 Bag of Visual Words with SIFT features

Let us apply these methods to the feature descriptors we received from the SIFT algorithm. These feature descriptors are vectors in an $\mathbb{R}^{128}$ space.

Due to a small diversity among vectors describing the same feature, we generate k centroids using the k-means method. These centroids represent the bins in our BoVW histogram.

For each picture, in order to generate a BoVW, the closest centroid for its feature descriptors is determined and assigned it to the centroids' cluster. The values in each bin in our BoVW will represent the amounts of feature descriptors in respective cluster.

4 Support Vector Machines

SVM algorithm was introduced by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963 [13]. It is the supervised learning classifier originally designed for binary classifications. Supervised learning means: the classifier generalizes the model from already categorized training samples. This model is then used to predict the group of unseen samples.

Let $T := \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), ..., (\boldsymbol{x_n}, y_n)\}$, be the training dataset, where n is the number of the samples, $\boldsymbol{x_i} \in \mathbb{R}^p$, $i \in \{1, 2, \ldots, n\}$, is the sample and $y_i \in \{-1, 1\}$ is the label related to the sample x_i.

The classification model is represented by the means of the hyperplane H, defined such that:

$$H : \omega^T x - \widetilde{b} = 0, \tag{21}$$

where ω is the normalized normal vector of the hyperplane H, and

$$\widetilde{b} = \frac{b}{\|\omega\|} \tag{22}$$

is the bias from the origin.

4.1 Hard-margin

Originally, the algorithm was designed for linearly separable training data. The two classes of data are distinguished by two hyperplanes so that the distance between them is maximised. The area between these hyperplanes is commonly called the margin and the maximal margin hyperplane H (21) lies between them in the middle. These hyperplanes are described by the following equations:

$$\boldsymbol{w}^T \boldsymbol{x} - \widetilde{b} = 1, \tag{23}$$

and

$$\boldsymbol{w}^T \boldsymbol{x} - \widetilde{b} = -1. \tag{24}$$

The data-point $\boldsymbol{x_i}$ belongs to a positive class, i.e. $y_i = 1$, when

$$\boldsymbol{w}^T \boldsymbol{x_i} - b \geq 1 \tag{25}$$

and in a negative class, i.e. $y_i = -1$, when

$$\boldsymbol{w}^T \boldsymbol{x_i} - b \leq -1. \tag{26}$$

For the convenience, (25) and (26) can be rewritten in following way:

$$y_i(\boldsymbol{w}^T \boldsymbol{x} - b) \geq 1. \tag{27}$$

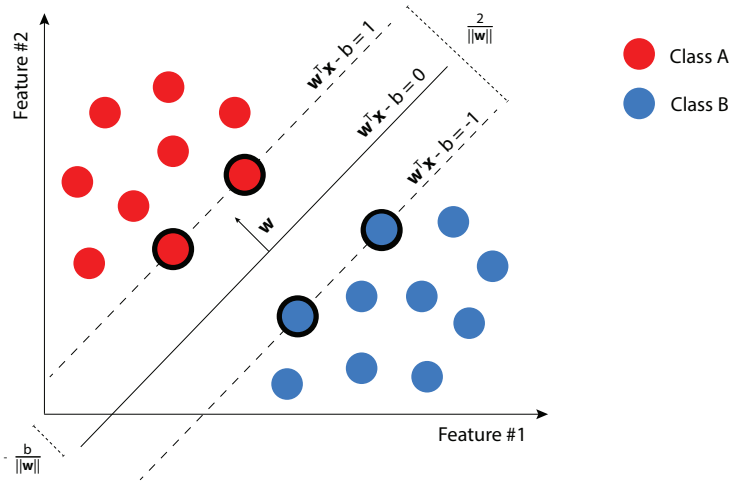


Figure 6: Example of the SVM model. [14]

From the Figure 6, we can see, the distance between hyperplanes (23) and (24) is $\frac{2}{\|\boldsymbol{w}\|}$. In the SVM model, the distance is maximized, therefore we minimize $\|\boldsymbol{w}\|$. Mathematically, it leads to an optimization problem

$$\arg\min_{\boldsymbol{w},b} \|\boldsymbol{w}\| \text{ s.t. } \begin{cases} y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) \geq 1, \\ i = 1, ..., n. \end{cases} \tag{28}$$

The optimization problem (28) can be reformulated as the Quadratic Programming (QP) problem:

$$\arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \text{ s.t. } \begin{cases} y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) \geq 1, \\ i = 1, ..., n. \end{cases} \tag{29}$$

The vectors $\boldsymbol{x_i}$, which lie on the hyperplanes (23) and (24), are called Support Vectors (SV).

4.2 Soft-margin

In 1993, the soft-margin version was proposed by Vladimir N. Vapnik and Corinna Cortes [Cortes1995]. It allows the SVM classifier to process non-linearly separable data exploiting of additional function called the hinge loss function:

$$\xi_i = \max(0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b)). \tag{30}$$

When a data-point lies on the correct side of the margin, the value of the hinge loss function equals to 0. For the data-point on the wrong side of the margin (see Figure 7), the value of the function is proportional to the distance from the margin.

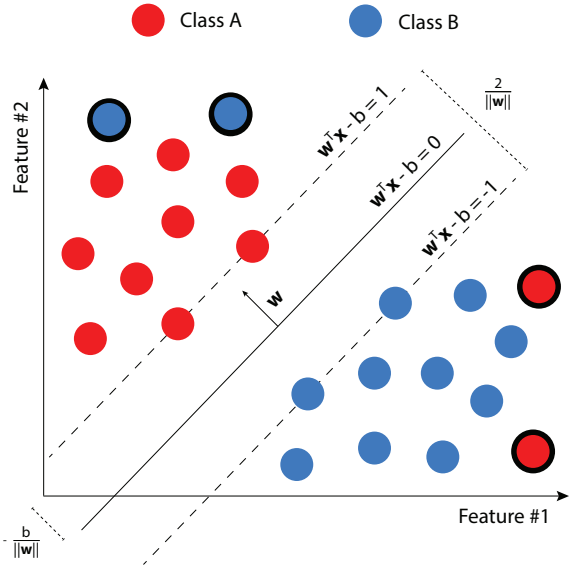


Figure 7: Example of a missclassified data-point. [14]

Adding the hinge loss function ξ to the optimization problem (29), we can obtain the following QP problem:

$$\arg\min_{\boldsymbol{w},b,\xi_i} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i \text{ s.t. } \begin{cases} y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) \geq 1 - \xi_i, \\ \xi_i \geq 0, i = 1, ..., n. \end{cases} \tag{31}$$

where C penalizes misclassification error. This formulation is often called the primal $l1$-loss SVM. By solving the (29) for Lagrange dual using Lagrange multipliers $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_3]$ and $\boldsymbol{\beta} = [\beta_1, \beta_2, ..., \beta_3]$, we obtain the dual formulation:

$$\arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{Y}^T\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\boldsymbol{e} \text{ s.t. } \begin{cases} \boldsymbol{o} \leq \boldsymbol{\alpha} \leq C\boldsymbol{e}, \\ \boldsymbol{B_e}\boldsymbol{\alpha} = 0, \end{cases} \tag{32}$$

where $\boldsymbol{e} = [1, 1, ..., 1]$, $\boldsymbol{o} = [0, 0, ..., 0]$, $\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}]$, $\boldsymbol{y} = [y_1, y_2, ..., y_n]$, $Y = \text{diag}(\boldsymbol{y})$, $\boldsymbol{B_e} = [\boldsymbol{y}^T]$ and $\boldsymbol{K} = \boldsymbol{X}^T\boldsymbol{X}$ is the Gram matrix [15]. The Hessian matrix in (32) is defined such that:

$$\boldsymbol{H} := \boldsymbol{Y}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{Y}. \tag{33}$$

In order to recover the normal vector, we use the following formula

$$\boldsymbol{w} = \boldsymbol{X}\boldsymbol{Y}\boldsymbol{\alpha}. \tag{34}$$

The bias b can be recovered by using $\bar{\boldsymbol{x}}$, a mean of all SV and evaluating

$$b = \boldsymbol{w} \cdot \bar{\boldsymbol{x}} - y_i. \tag{35}$$

Using the square sum of the loss functions ξ_i instead of a linear sum, in $l1$-loss SVM (31), we get the $l2$-loss SVM as follows:

$$\arg\min_{\boldsymbol{w},b,\xi_i} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 \text{ s.t. } \begin{cases} y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) \geq 1 - \xi_i, \\ i = 1, ..., n. \end{cases} \tag{36}$$

Same as in case of $l1$-loss SVM, we can derive dual formulation using the Lagrange duality, resulting in

$$\arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^T(\boldsymbol{H} + C^{-1}\boldsymbol{I})\boldsymbol{\alpha} - \boldsymbol{\alpha}^T e \text{ s.t. } \begin{cases} \boldsymbol{o} \leq \boldsymbol{\alpha}, \\ \boldsymbol{B_e}\boldsymbol{\alpha} = 0. \end{cases} \tag{37}$$

Because the Hessian matrix $\boldsymbol{H}$ regularized by matrix $C^{-1}\boldsymbol{I}$ is symmetric positive definite, this optimization problem should be mathematically more stable than in the case of the $l1$-loss SVM.

4.3 No bias data classification

For sparse data in a high dimensional space, the bias term b in primal formulation is not needed [15]. Therefore, it can be avoided from from primal formulations of SVM problems.

Then, the primal $l1$-loss SVM from (31) and the $l2$-loss l2-regularized SVM from (36) result in no bias formulations, so that

$$\arg\min_{\boldsymbol{w},\xi_i} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i \text{ s.t. } \begin{cases} y_i(\boldsymbol{w}^T\boldsymbol{x}_i) \geq 1 - \xi_i, \\ \xi_i \geq 0, \\ i = 1, ..., n, \end{cases} \tag{38}$$

and

$$\arg\min_{\boldsymbol{w},\xi_i} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 \text{ s.t. } \begin{cases} y_i(\boldsymbol{w}^T\boldsymbol{x}_i) \geq 1 - \xi_i, \\ \xi_i \geq 0, \\ i = 1, ..., n, \end{cases} \tag{39}$$

respectively. By applying Lagrange mmultipliers , the dual formulation in (32) becomes

$$\arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{Y}^T\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\alpha} - \boldsymbol{\alpha}^T\boldsymbol{e} \text{ s.t. } \boldsymbol{o} \leq \boldsymbol{\alpha} \leq C\boldsymbol{e}, \tag{40}$$

and the dual formulation in (37) becomes

$$\arg\min_{\boldsymbol{\alpha}} \frac{1}{2}\boldsymbol{\alpha}^T(\boldsymbol{H} + C^{-1}\boldsymbol{I})\boldsymbol{\alpha} - \boldsymbol{\alpha}^T e \text{ s.t. } \boldsymbol{o} \leq \boldsymbol{\alpha}. \tag{41}$$

In some cases, these OPs produce a poor classification model. we can prevent it by appending an additional feature to each sample. The additional feature is usually set to 1. Using these extended samples, we rewrite the separating hyperplane equation in component-wise form as

follows:

$$w_1x_1 + w_2x_2 + ... + w_px_p + 1w_{p+1} = 0, \tag{42}$$

where the first p members can be seen as $\boldsymbol{w}^T\boldsymbol{x}$ and the last member can be seen as the bias b. From (42), we can see that hyperplanes related to the non-bias formulations (40) and (41) are equivalent to the hyperplanes related the original formulations (32).

4.4 Grid search

SVM, same as many other learning algorithms has some parameters, for example a penalty for misclassification error C in (31), which can affect the classification performance. There are many different approaches to finding the right value for these parameters. One of these approaches is a Grid search.

Grid search (GS) is an exhaustive search through a provided set of parameter values. The SVM is trained with each of these parameter value combinations. In order to measure the classification performance with given parameters, a cross-validation technique is used.

4.5 Cross-validation

It is important to assess the performance of the model on a new independent data, data that has not been used for the training of the model. To achieve this, a stratified k-fold cross-validation technique is used.

First, let us look at a k-fold cross-validation. It is a non-exhaustive, i.e. not all possible ways of splitting data are used, cross validation technique. In this approach, the original sample set is split into k subsets (folds) of an equal size. One of the folds is kept as a validation data and the model is trained on the other $k-1$ folds. This process is repeated k times, always picking a different fold as a validation data. The results of all k runs are then averaged to give a single estimation. In this approach, every data point is used as a validation data exactly once.

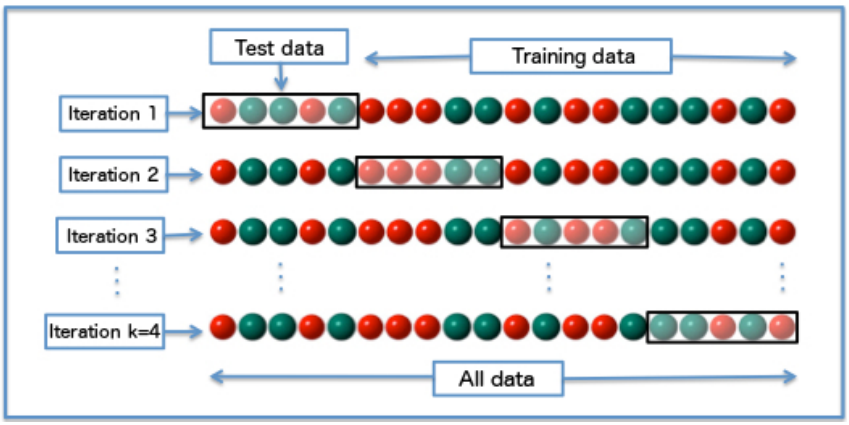


Figure 8: k-fold CV splits data into k folds and trains the model k times always leaving out a different fold as a validation set. [16]

A randomly selected fold may not represent the classes in the same ratio as the whole set. Stratified k-fold cross-validation ensures that the ratios of the classes in the folds are roughly equal, i.e. if we have 50% of data in a positive class and 50% in the negative class, this method ensures that we have the classes represented in same percentages in each of the folds.

Further in the text, we are referencing the stratified k-fold crossvalidation as just a cross-validation or CV.

4.6 Metrics

In order to evaluate the performance core of the classification model generated by SVM, we will be analysing its confusion matrix (see Table 1).

	Predicted negative	Predicted positive
Actual negative	True negatives (TN)	False positives (FP)
Actual positive	False negatives (FN)	True positives (TP)

Table 1: Confusion Matrix

From the matrix, we can generate the following metrics:

Accuracy: How often is the classifier correct overall. This metric is represented in percentages.

$$\frac{TN + TP}{TN + FP + FN + TP} * 100[\%] \tag{43}$$

Precision: How often is the classifier correct when it predicts positive.

$$\frac{TP}{TP + FP} * 100[\%] \tag{44}$$

Sensitivity (Recall): How often is the classifier correct when it is actually positive.

$$\frac{TP}{TP + FN} * 100[\%] \tag{45}$$

F_1 **Score:** A harmonic mean of precision and sensitivity.

$$2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{46}$$

AUC ROC: AUC ROC, an area under a Receiver Operating Characteristic (ROC) curve. ROC curve represents precision plotted against a False Positive Rate, a probability of wrongly predicting when it is actually negative at various biases b. The larger the area under this curve the better classifier we have.

All of these metrics can be from $< 0, 1 >$. The values around 0.5 mean that our classifier is not better than a random classifier and the higher the value the better.

5 Tools

There are few steps, we need to do when we want to use a dataset for feature extraction and classification. The data has to be analysed, labeled and sometimes rearranged. For the preparation, feature extraction, and classification, we use the following tools.

5.1 Python Libraries

In the benchmarks we use the following python libraries for the dataset preparation and FE:

OpenCV [17] Open Source Computer Vision Library (OpenCV) is an computer vision and machine learning software library. It provides a C++, Python and Java interfaces. It contains thousands of algorithms used for machine learning, image manipulation and much more. Among others, OpenCV has an implementation of a SIFT algorithm and k-means algorithm, both of which are useful in our benchmarks.

NumPy [18] A Python library, used mainly for array manipulation.

Scikit-learn [19] A Python machine learning library. It contains algorithms for calculating a Calinski-Harabasz and Davies-Bouldin Indexes, which we use.

5.2 PermonSVM

The tool used for training and testing of the classification model is PermonSVM [20]. It is an SVM classifier implementation developed at Department of Applied Mathematics, VSB-TUO, and Institute of Geonics of the Czech Academy of Science in Ostrava, Czech Republic. It is built on top of PermonQP [21], a package for large scale quadratic programming (QP) problems. Both make use of PETSc [22], a portable, extensible toolkit for scientific computation.

PermonSVM utilizes an implicit representation of the matrix product $\boldsymbol{X}^T\boldsymbol{X}$ in order to save memory and CPU time. The quadratic programming problem is solved using the QP solver from PermonQP [21].

This SVM classifier implementation provides a confusion matrix and scores for numerous different metrics, including accuracy, precision, sensitivity, F_1 score and AUC ROC. It can perform a stratified cross validation and a grid search, i.e. searching for the best parameter values. PermonSVM is also capable of setting a penalty for unbalanced dastasets [15].

It is able to load the datasets from an HDF5 file format.

6 Dataset Preparations

For the benchmarks in Section 7, we use the IIIT Pet Dataset [23]. It contains more than 7000 images of different cat and dog breeds, see examples in Figure 9. The animals are photographed in different conditions (light condition, taken from different point of view, etc.), environments and the images have various resolutions. The dataset includes trimaps of the most images, allowing us to cut out only the parts of the image scene, where the animals are depicted, for the pets classification in Section 7.4. Moreover, it provides a bounding boxes surrounding the animal faces for most of the images. This is used for the pet faces classification in Section 7.2.

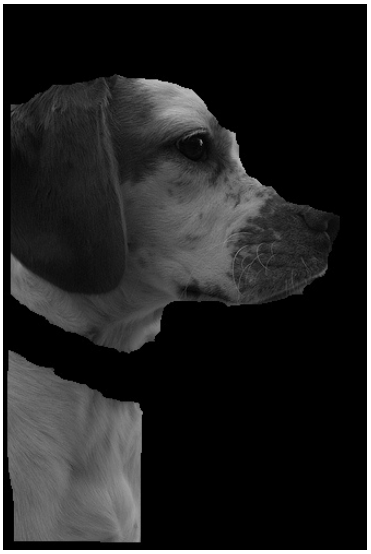
Figure 9: Examples of images depicting cat and dog in the IIIT Pet Dataset. [23]

Despite having more than 7000 images, the data annotations are available only for some of them. Moreover, the number of the dog images is more than twice the number of cat images. Therefore, we avoid all images, for which the data annotations are missing, and we reduce the number of the dog images to half, so that we get approximately balanced dataset. Then, the pixel values in the image backgrounds are set to zero. We label the images according to their classes: cats being the class $+1$ and dogs being the class -1. The $\frac{1}{4}$ of each class is used as the testing data. The resulting dataset is referenced as Pets dataset further in the thesis. From the Pets dataset, we create the Pets Faces dataset by cropping the images around the face using the bounding boxes mentioned above. The example of the original dataset, its Pets dataset counterpart and Pets Faces dataset counterpart is shown in Figure 10. In order to simplify the illumination based classification, the outliers in resolution are discarded from both of our datasets.

6.1 Illumination Based Classification Dataset

For the illumination based classification, the images have to be resized to the same resolution. To achieve this, we add columns of zeros to each image with the smaller width than the maximum one. The same is done for the height using rows of zeros. In the case of the Pets Faces, the width and height of reshaped images are 400px and 400px, respectively, and the resolution is

(a) Original image of a beagle.



(b) A cutout of the beagle.

(c) Cropped face of the beagle.

Figure 10: An image of a beagle in the original, pets, and Pets Faces dataset. [23]

700 × 700px in the case of the Pets dataset. Then, the images are reshaped into vectors with dimension 160,000 and 490,000 for the Pets Faces and Pets datasets, respectively. Then we store the training and test data their labels into HDF5 files.

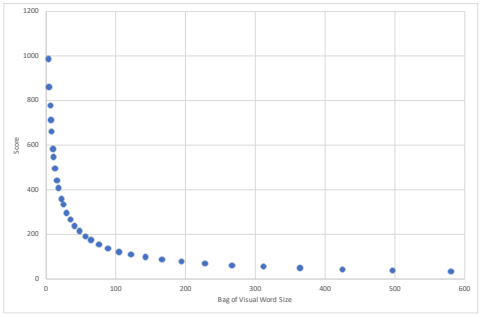
6.2 SIFT Features Dataset

For the classification using the SIFT features, the features are extracted from each image using the SIFT implementation in an OpenCV library (mentioned in the Section 5). Then, the BoVW is created and the original data is represented exploiting it. The process is described in the Section 3. We use the same format, i.e. an HDF5 file, to store the new representation of the images and the class labels. The extraction is performed on both the Pets and the Pets Faces datasets.

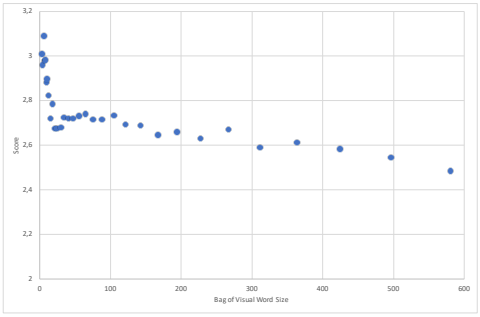
The biggest challenge of k-means clustering is to determine the number of clusters. Firstly, we try to achieve it by using the k-means clustering with various number of clusters. Then, the results are evaluated employing Calinski-Harabaz and Davies-Bouldin indexes.

Unfortunately, as we can see in Figure 11 for the Pets dataset and in Figure 12 for the Pets Faces dataset, these metrics contradict each other. The Calinski-Harabaz index indicates better clustering for the lower values of clusters while the Davies-Bouldin indicates better clustering for the higher values of clusters.

The number of clusters can also be determined experimentally. For each dataset, nine variations are generated with the number of clusters $k \in \{2, 4, 8, 16, 32, 64, 128, 256, 512\}$. We train the SVM model for each of the variation and compare the results.

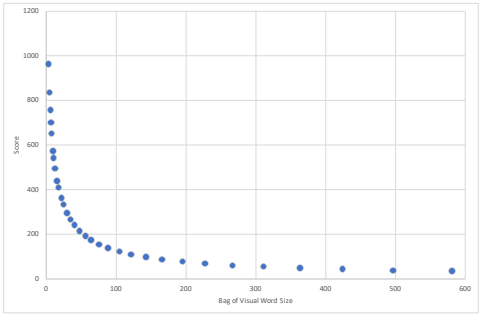


(a) Calinski-Harabaz index evolution for different BoVW sizes.

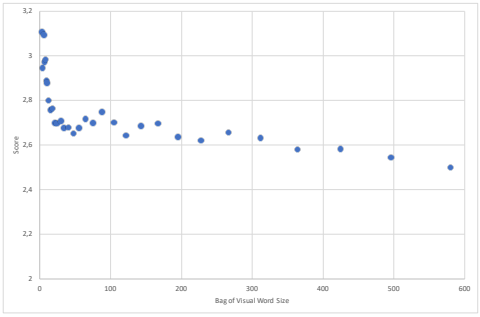


(b) Davies-Bouldin index evolution for different BoVW sizes.

Figure 11: Metrics comparing different Bag of Visual Words dictionary sizes for the Pets dataset.



(a) Calinski-Harabaz index evolution for different BoVW sizes.



(b) Davies-Bouldin index evolution for different BoVW sizes.

Figure 12: Metrics comparing different Bag of Visual Words dictionary sizes for the Pets Faces dataset.

7 Benchmarks

In this section, we compare the illumination based classification with the classification using BoVW of the SIFT feature vectors on Pets Faces and Pets datasets mentioned in Section 6. We run the presented experiments on the Salomon supercomputer [24] at IT4Innovations. Salomon consists of 1008 compute nodes. Each compute node contains two 2.5 GHz, 12-core Intel Xeon E5-2680v3 (Haswell) processors and 128 GB of memory. Compute nodes are interconnected by InfiniBand FDR56. Salomon has the peak performance of 2 petaFLOPS. As the underlying SVM solver, we employ the PermonSVM, see Section 5.2 for further information. We choose the best penalty C_{Best} from

$$C = \{2^i, i \in \{-10, -9, \ldots, 10\}\} \tag{47}$$

algorithmically using the hyperparameter optimization (HyperOpt) by means of GS combined with 5-fold CV. The C_{Best} was determined as $\widehat{C}$ from (47) such that the related mean of the accumulated precision, sensitivity and AUC ROC metrics is maximal. We use the Modified Proportioning and Reduced Gradient Projection (MPRGP) [25] as the underlying solver for the QP problems arising from dual $l1$-loss SVM and $l2$-loss SVM formulations. The experiments run on 16 MPI processes in case of the illumination based classification, and experiments associated with classification using the SIFT features run on 1 MPI process.

7.1 Pets Faces: Illumination Based Classification

In this experiment, we classify the raw data of the Pets Faces dataset, i.e. illumination based classification. The selected penalty C_{Best}, the SVM model performance scores, namely F1 and Sensitivity, related to the test dataset, the elapsed time and the number of iterations are summarised in Table 2 and in Table 3 for $l1$-loss SVM and $l2$-loss SVM, respectively.

Model			#Iterations	HyperOpt + Training
C_{Best}	F1	Sens. [%]	(training)	[hh:mm:ss]
2^{-10}	0.76	75.43	559	13:34:09.43

Table 2: Illumination based Pets Faces dataset classification ($l1$-loss SVM): Performance scores of the model associated with C_{Best} evaluated on test dataset, number of iterations related to the training procedure, elapsed time.

From the tables Table 2 and Table 3, we observe, the models associated with the $l1$-loss SVM and $l2$-loss SVM are well trained. The $l2$-loss SVM performed slightly better according to the scores: the difference of F1 scores is 0.02 and 2.1% in the case of the sensitivity. Furthermore, the computational time for HyperOpt and training the $l2$-loss SVM is 41 minutes and 28 seconds faster and converges in less iterations, i.e 168 iterations in difference.

Model			#Iterations (training)	HyperOpt + Training [hh:mm:ss]
C_{Best}	F1	Sens. [%]		
2^{-10}	0.78	77.39	391	12:52:41.15

Table 3: Illumination based Pets Faces dataset classification ($l2$-loss SVM): Performance scores of the model associated with C_{Best} evaluated on test dataset, number of iterations related to the training procedure, elapsed time.

7.2 Pets Faces: Classification using Extracted SIFT Features

This experiment deals with exploiting the SIFT descriptors (feature vectors) for improving classification. We demonstrate the impact of using the descriptors instead of the pixel intensities, see Section 7.1, on Pets Faces dataset. In this benchmark, we monitor the training and test scores, namely sensitivity and F1, during hyperparameter optimization associated with different BoVW sizes for both $l1$-loss and $l2$-loss SVM as well; the achieved results are presented and visualized as the boxplots. The BoVW sizes are chosen from $k_{BoVW} = \{2^i : i \in 1, 2, \dots 9\}$.

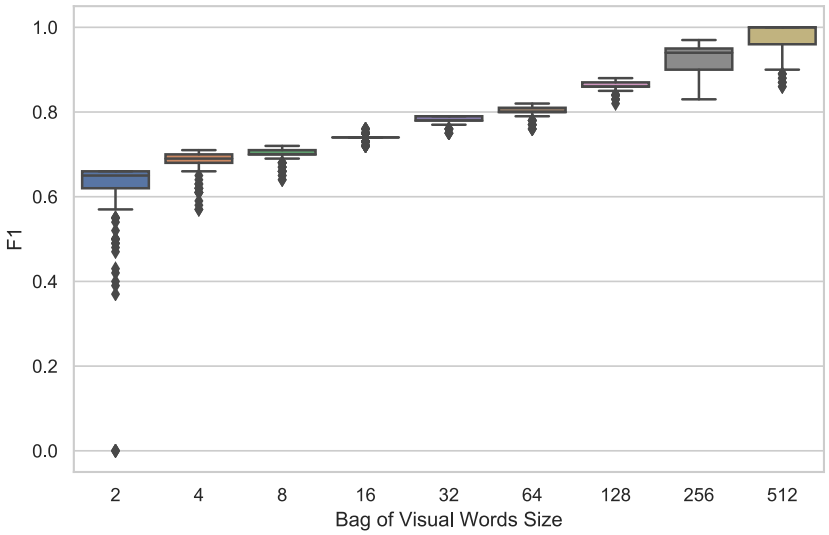
Analyzing the Figure 13 and Figure 14, we can observe, the increase of F1 score and sensitivity along with the increase of the BoVW dictionary size for both $l1$-loss SVM and $l2$-loss SVM. For the BoVW dictionary size 512, the median F1 score is 1 and the median sensitivity 100% on training dataset. It could cause overfitting, i.e. classification model that fits too closely to training dataset; it may result in poor performance on unseen data. Therefore, we analyze the scores achieved during HyperOpt on testing data, see Figure 15 and Figure 16 for F1 and sensitivity, respectively.

In Figure 15 and Figure 16, we observe, the median scores achieved during HyperOpt on testing dataset indicate a good classifier for 16 or more clusters; the median F1 score is above 0.70 and the median accuracy is above 70% for both $l1$-loss and $l2$-loss.

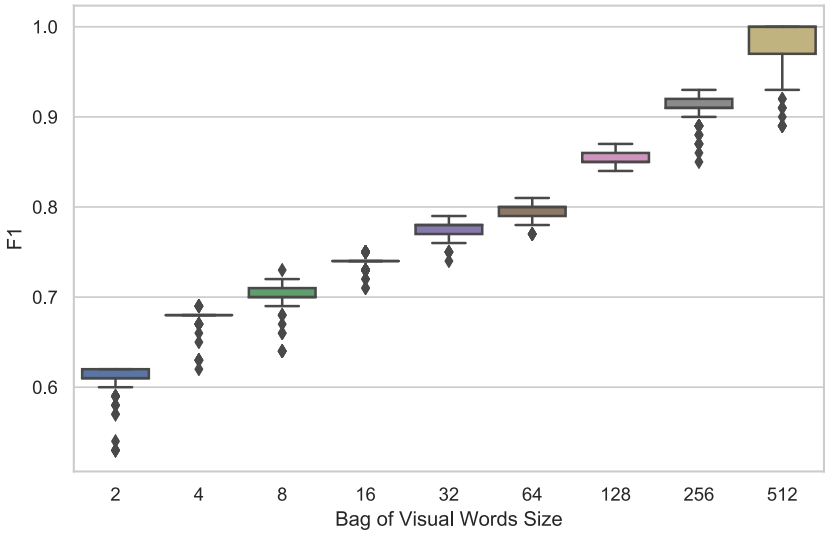
The best penalty C_{Best} is selected and the model is trained using the C_{Best} for each number of clusters.

BoVW size	Model			#Iterations (training)	Elapsed times [hh:mm:ss]		
	C_{Best}	F1	Sens. [%]		FE	HyperOpt + Training	$\sum$
2	2^1	0.62	49.87	2839	0:00:05:23	0:00:00:21	0:00:05:45
4	2^5	0.65	54.63	53187	0:30:32.05	0:00:24.40	0:30:56.45
8	2^3	0.62	55.25	27904	1:21:59.83	0:00:29.40	1:22:29.23
16	2^5	0.67	61.68	128096	2:43:12.36	0:00:37.13	2:43:49.49
32	2^6	0.70	67.61	124732	1:52:20.52	0:01:03.73	1:53:24.25
64	2^3	0.77	74.79	7455	1:36:05.62	0:01:50.33	1:37:55.95
128	2^2	0.75	71.49	2274	0:57:58.32	0:03:06.48	1:01:04.80
256	2^{-9}	0.75	79.31	194	0:11:11.86	0:06:40.05	0:17:51.91
512	2^{-10}	0.82	85.92	60	0:02:00.11	0:12:38.13	0:14:38.24

Table 4: Pets Faces dataset using SIFT ($l1$-loss SVM): Classification model evaluated on test data for different BoVW dictionary sizes.

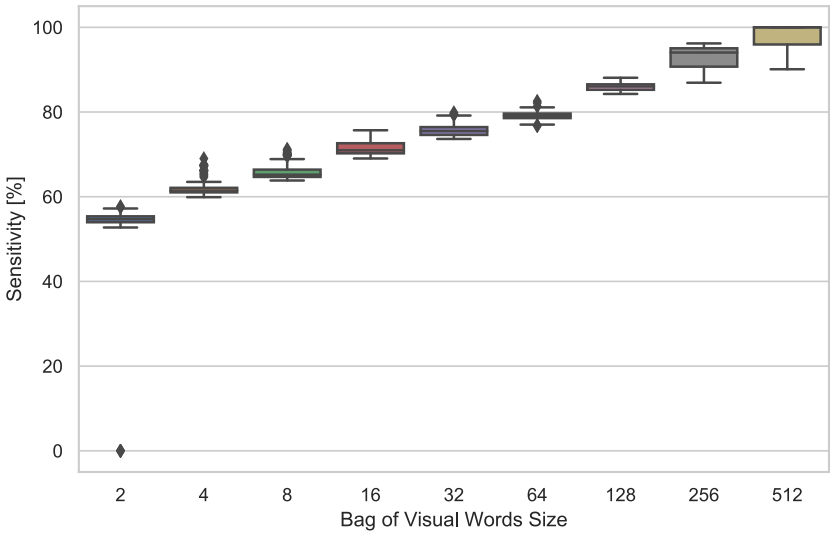


(a) $l1$-loss SVM: the F1 training scores.

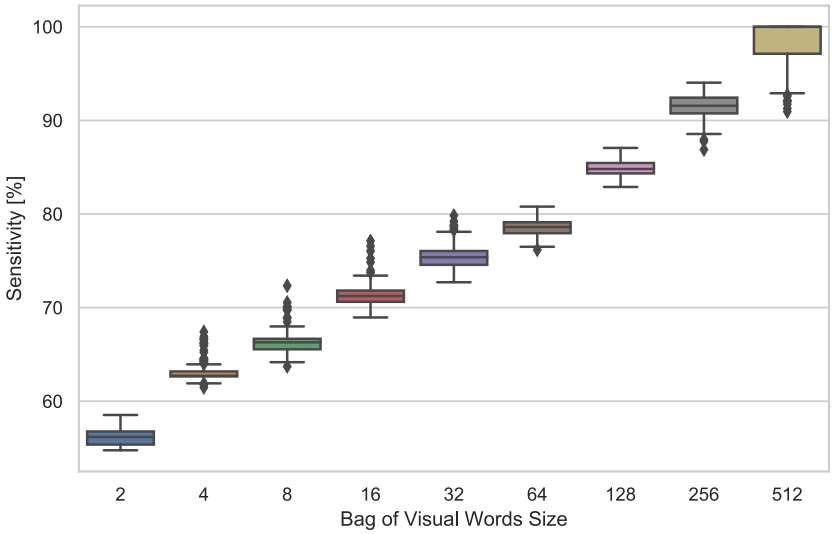


(b) $l2$-loss SVM: the F1 training scores.

Figure 13: F1 training scores during the HyperOpt on Pets Faces dataset for different BoVW sizes.

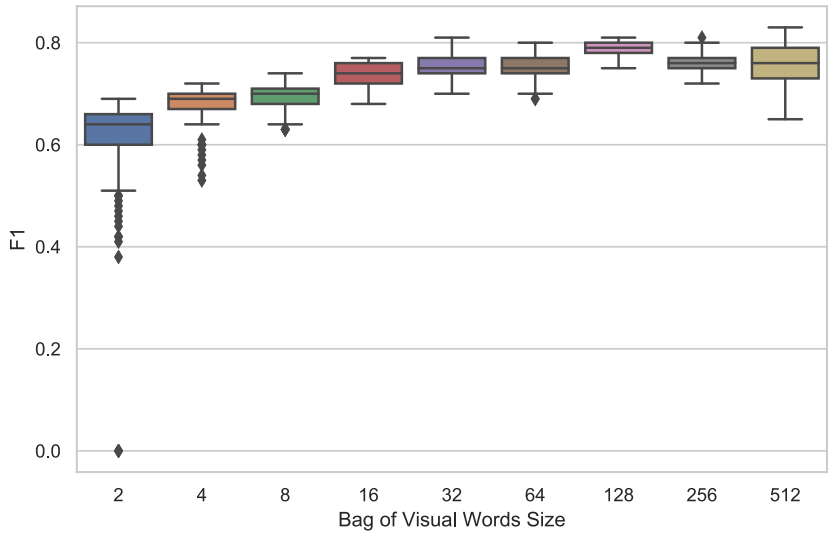


(a) $l1$-loss SVM: the sensitivity training scores.

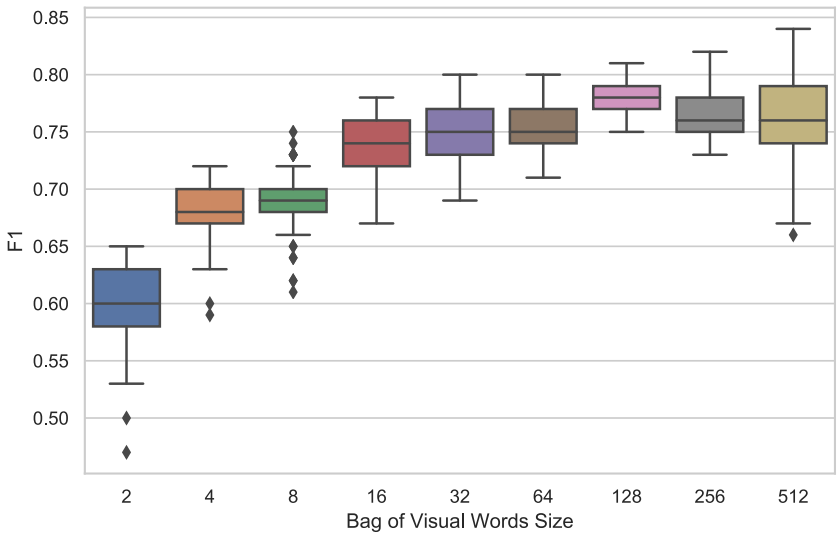


(b) $l2$-loss SVM: the sensitivity training scores.

Figure 14: Sensitivity training scores during the HyperOpt on Pets Faces dataset for different BoVW sizes.

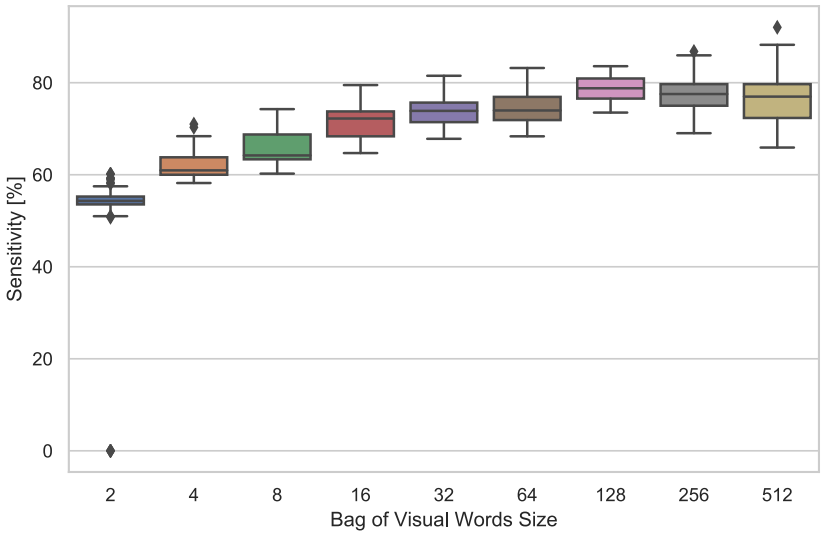


(a) $l1$-loss SVM: The F1 test scores.

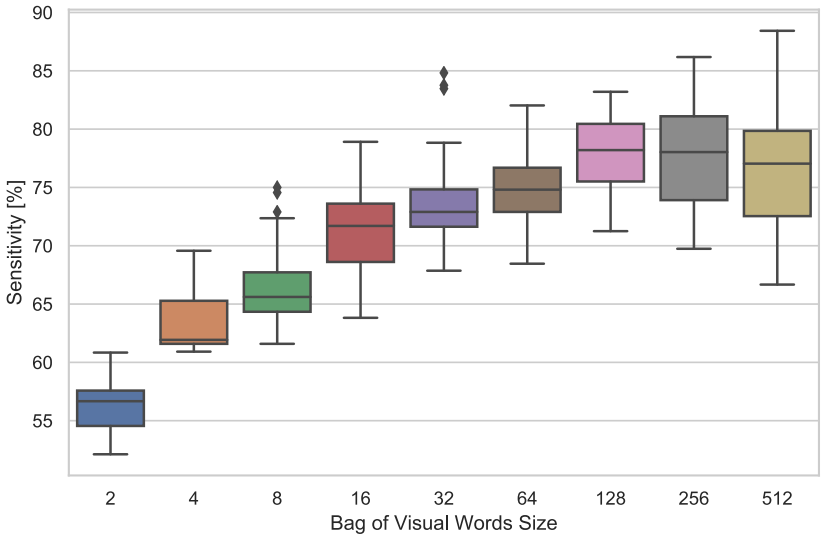


(b) $l2$-loss SVM: the F1 test scores.

Figure 15: F1 test scores during the HyperOpt on Pets Faces dataset for different BoVW sizes.



(a) $l1$-loss SVM: the sensitivity test scores.



(b) $l2$-loss SVM: the sensitivity test scores.

Figure 16: Sensitivity test scores during the HyperOpt on Pets Faces dataset for different BoVW sizes.

BoVW size	Model			#Iterations (training)	Elapsed times [hh:mm:ss]		
	C_{Best}	F1	Sens. [%]		FE	HyperOpt + Training	$\sum$
2	2^{10}	0.57	49.84	3	0:00:00.19	0:00:21.94	0:00:22.13
4	2^{-6}	0.63	56.84	67	0:00:02.77	0:00:24.40	0:00:27.17
8	2^{-1}	0.63	56.34	159	0:00:05.25	0:00:29.40	0:00:34.65
16	2^{-3}	0.66	62.16	181	0:00:09.40	0:00:37.13	0:00:46.53
32	2^{-5}	0.70	67.21	192	0:00:13.50	0:01:03.73	0:01:17.23
64	2^{4}	0.77	75.85	257	0:00:22.18	0:01:50.33	0:02:12.51
128	2^{-10}	0.73	74.32	67	0:00:47.70	0:03:06.48	0:03:54.18
256	2^{-10}	0.76	78.67	45	0:01:40.01	0:06:40.05	0:08:20.06
512	2^{-10}	0.80	83.41	29	0:01:51.38	0:12:38.13	0:14:29.51

Table 5: Pets Faces dataset using SIFT ($l2$-loss SVM): Classification model evaluated on test data for different BoVW dictionary sizes.

In the tables Table 4 and Table 5, we observe the SVM model performance scores with the selected penalty C for each BoVW dictionary size. In addition, the tables display the number of iterations, the elapsed time of HyperOpt and training and the elapsed time of features extraction and total elapsed time.

Looking at the table Table 4 we observe high number of iterations for BoVW dictionary sizes 16 and 32. Therefore, the solution could be affected by enormous numerical error. The models for the BoVW dictionary sizes from 64 to 512 are considered good classifiers for both $l1$-loss and $l2$-loss SVM.

We get the best classification scores for the BoVW size 512. The total computational times for both models with 512 clusters are similar, 14 minutes and 38 seconds for the $l1$-loss SVM and 14 minutes 29 seconds for the $l2$-loss SVM. The $l1$-loss SVM performed slightly better according to the scores: the difference for F1 is 0.2 and 2.51% in the case of sensitivity. The confusion matrices for $l1$-loss SVM and $l2$-loss SVM can be seen in Figure 10 an Figure 11, respectively.

Compared to the results of the illumination based classification in Section 7.1, the classification using the SIFT features provides small improvement in the performance scores. The classifier for the illumination based classification achieved 0.78 and 77.39% for F1 and sensitivity respectively. Using SIFT features, we were able to achieve an improvement of 0.04 for the F1 score and 8.51% for the sensitivity. Moreover, the total elapsed time for the classification using the SIFT features is order of magnitude shorter (under 15 minutes compared to nearly 13 hours). This is due to a data dimension reduction using SIFT extraction. This experiment shows us that the SIFT FE is useful in combination with SVM for image classification.

7.3 Pets: Illumination Based Classification

In the next experiment, we classify the raw data of the Pets dataset. The selected penalty C_{Best}, the SVM model performance scores, namely F1 and Sensitivity, related to the test dataset, the

elapsed time and the number of iterations are summarised in Table 6 and in Table 7 for $l1$-loss SVM and $l2$-loss SVM, respectively.

Model			#Iterations (training)	HyperOpt + Training [hh:mm:ss]
C_{Best}	F1	Sensitivity [%]		
2^{-4}	0.57	56.03	2164	26:43:14.06

Table 6: Illumination based Pets dataset classification ($l1$-loss SVM): Performance scores of the model associated with C_{Best} evaluated on test dataset, number of iterations related to the training procedure, elapsed time.

Model			#Iterations (training)	HyperOpt + Training [hh:mm:ss]
C_{Best}	F1	Sensitivity [%]		
2^{-9}	0.55	54.84	516	24:25:33.21

Table 7: Illumination based Pets dataset classification ($l2$-loss SVM): Performance scores of the model associated with C_{Best} evaluated on test dataset, number of iterations related to the training procedure, elapsed time.

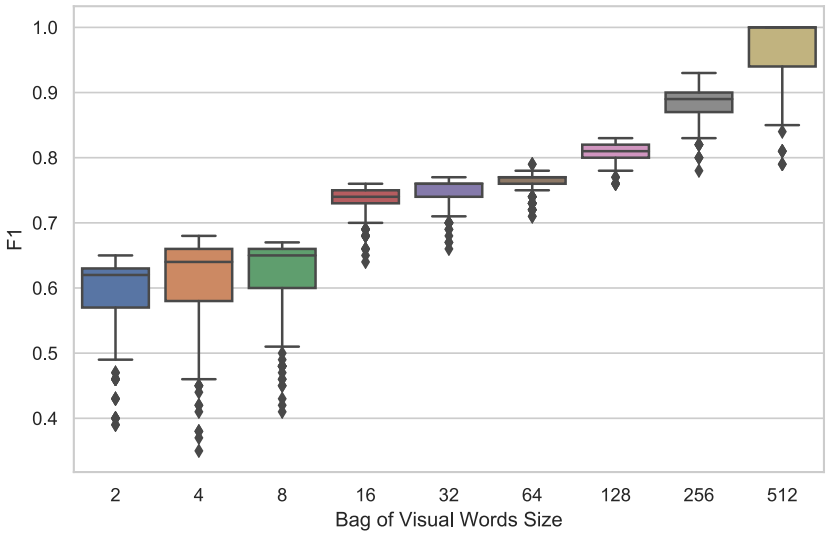
From the tables Table 6 and Table 7, we observe, both models perform poorly. Although, the model associated with the $l1$-loss SVM performed slightly better according to the scores: the difference of F1 scores is 0.02 and 1.29% in the case of the sensitivity. However, neither of the models classify the data much better than a random classifier.

7.4 Pets: Classification using Extracted SIFT Features

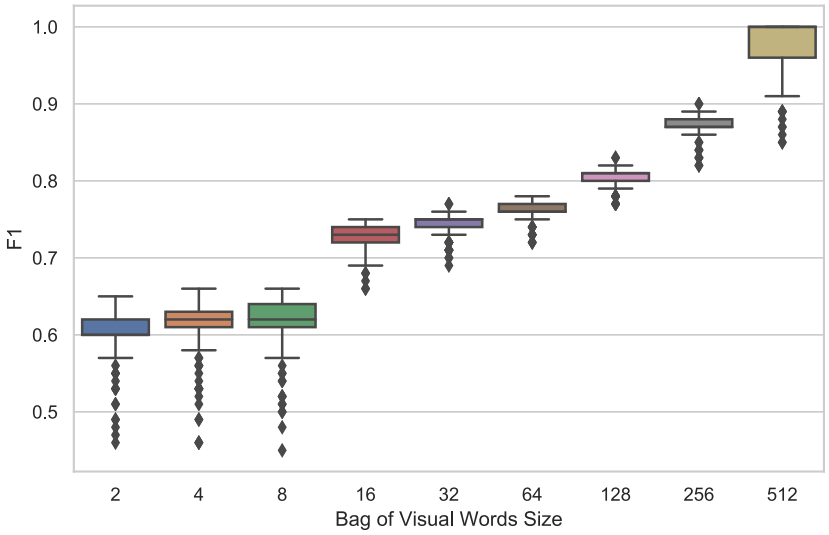
In this experiment we exploit the SIFT descriptors to improve the SVM model for the Pets dataset. We demonstrate the advantages of classification using SIFT features compared to the illumination based classification, see Section 7.3, on Pets Dataset. In the benchmark, we monitor the training and test scores, namely sensitivity and F1, during hyperparameter optimization associated with different BoVW sizes for both $l1$-loss and $l2$-loss SVM as well; the achieved results are represented and visualized as the boxplots. The BoVW sizes are chosen from k_{BoVW}, such that:

$$k_{BoVW} := \{2^i : i \in 1, 2, \dots 9\}. \tag{48}$$

Analyzing the Figure 17 and Figure 18, we observe a stagnation of the scores for the BoVW dictionary sizes 2, 4 and 8. There is increase of F1 score and sensitivity along with the increase of the BoVW dictionary size further from the size 16 for both $l1$-loss SVM and $l2$-loss SVM. For the BoVW dictionary size 512, the median F1 score is 1 and the median sensitivity 100% on training dataset. It could cause overfitting, which might result in poor performance on new data. Therefore, we analyze the scores achieved during HyperOpt on testing data, see Figure 19 and Figure 20 for F1 and sensitivity, respectively.

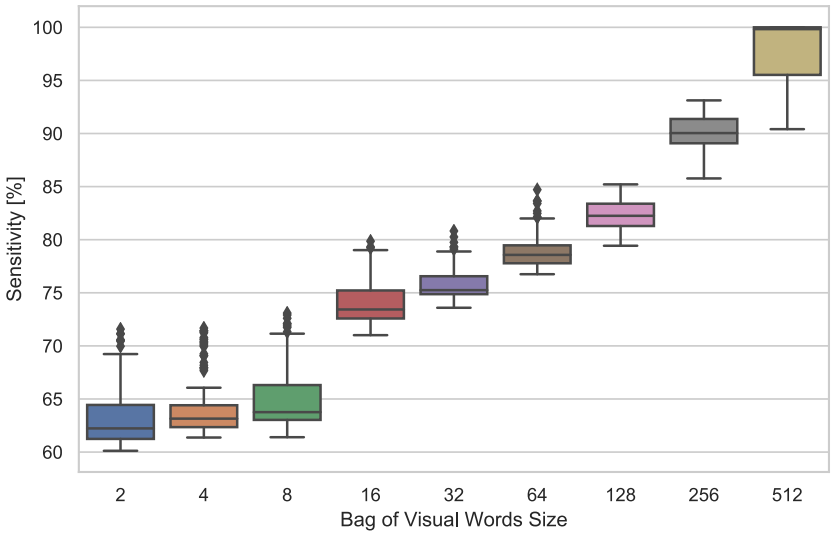


(a) $l1$-loss SVM: the F1 training scores.

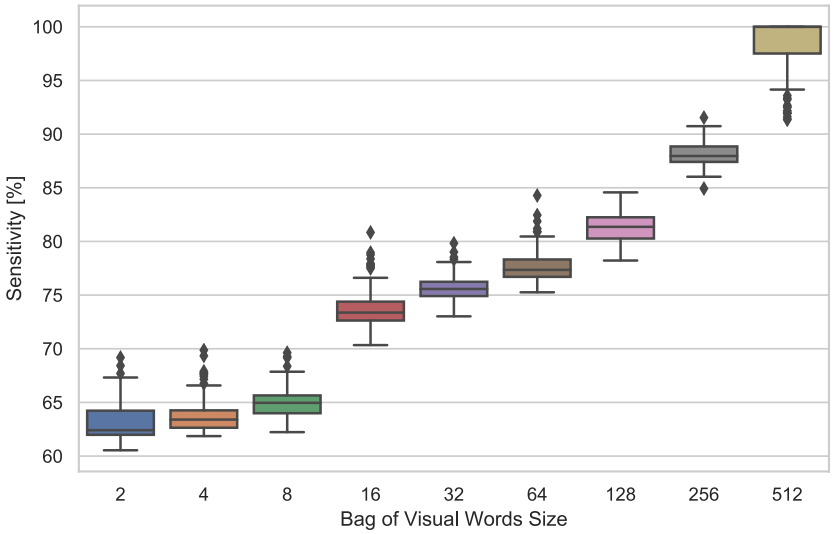


(b) $l2$-loss SVM: the F1 training scores.

Figure 17: F1 training scores during the HyperOpt of Pets dataset for different BoVW sizes.

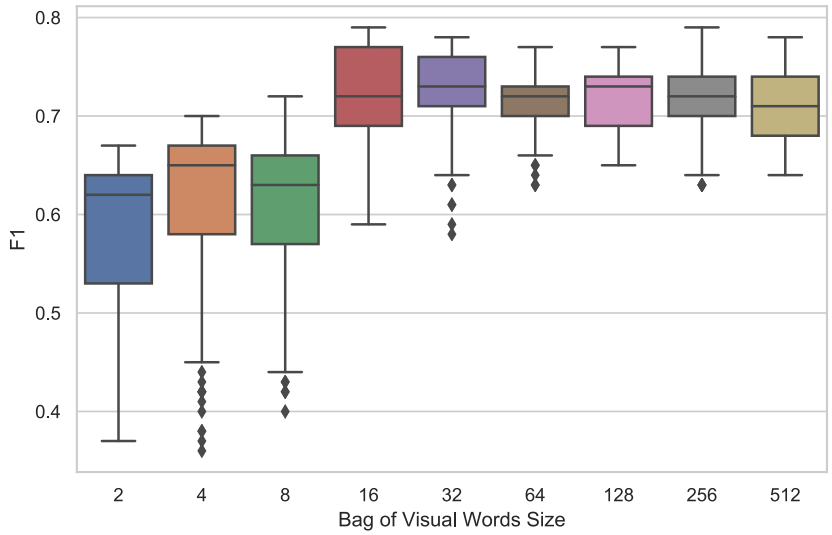


(a) $l1$-loss SVM: the sensitivity training scores.

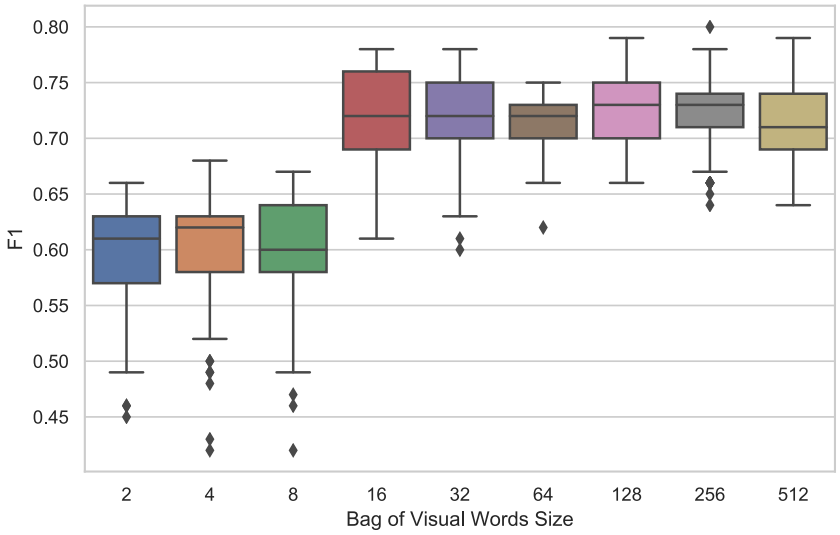


(b) $l2$-loss SVM: the sensitivity training scores.

Figure 18: Sensitivity training scores during the HyperOpt of Pets dataset for different BoVW sizes.

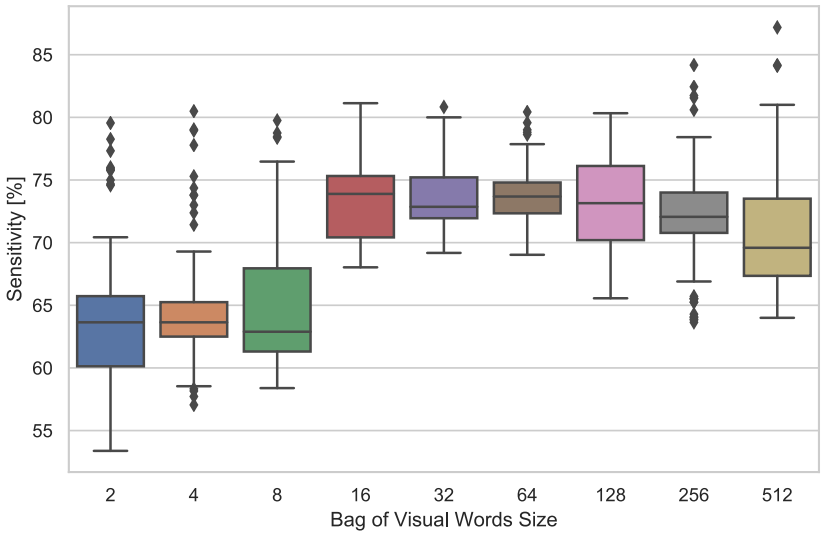


(a) $l1$-loss SVM: the F1 test scores.

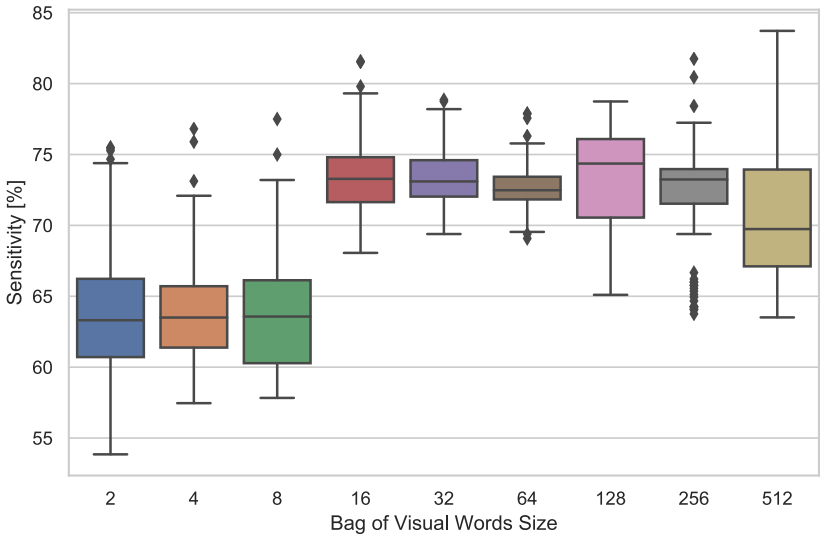


(b) $l2$-loss SVM: the F1 test scores.

Figure 19: F1 test scores during the HyperOpt of Pets dataset for different BoVW sizes.



(a) $l1$-loss SVM: the sensitivity test scores.



(b) $l2$-loss SVM: the sensitivity test scores.

Figure 20: Sensitivity test scores during the HyperOpt of Pets dataset for different BoVW sizes.

In Figure 20 and Figure 19, we can see a good classifiers are the ones for BoVW size 16 and above. To choose the best size, let us look at the Table 4 and Table 5, where we can see the penalty C, selected by HyperOpt, the F1 score and sensitivity on testing data, the number of iterations for the training with selected penalty, the elapsed time of training of SVM model including HyperOpt and the elapsed time of extraction for each BoVW dictionary size.

BoVW size	Model			#Iterations (training)	Elapsed times [hh:mm:ss]		
	C_{Best}	F1	Sens. [%]		FE	HyperOpt + Training	$\sum$
2	2^{4}	0.62	61.11	8663	0:10:51.96	0:01:00.39	0:11:52.35
4	2^{7}	0.65	60.78	157191	0:54:29.37	0:01:14.47	0:55:43.84
8	2^{2}	0.64	61.19	34507	2:20:25.92	0:01:18.02	2:21:43.94
16	2^{5}	0.69	70.39	306628	5:56:55.27	0:01:38.48	5:58:33.75
32	2^{1}	0.71	72.65	23144	7:00:26.73	0:02:15.13	7:02:41.86
64	2^{0}	0.73	72.47	5753	5:30:01.91	0:03:36.31	5:33:38.22
128	2^{2}	0.74	75.53	7256	2:39:10.83	0:06:01.50	2:45:12.33
256	2^{-8}	0.73	77.98	441	1:01:32.85	0:10:51.64	1:12:24.49
512	2^{-9}	0.74	79.44	235	0:04:27.94	0:20:20.61	0:24:48.55

Table 8: Pets dataset with SIFT ($l1$-loss SVM): Classification model evaluated on test data for different BoVW dictionary sizes.

BoVW size	Model			#Iterations (training)	Elapsed times [hh:mm:ss]		
	C_{Best}	F1	Sens. [%]		FE	HyperOpt + Training	$\sum$
2	2^{-3}	0.60	62.28	11	0:00:00.46	0:01:00.39	0:01:00.85
4	2^{-4}	0.63	60.92	39	0:00:01.35	0:01:14.47	0:01:15.82
8	2^{-1}	0.61	60.73	85	0:00:03.21	0:01:18.02	0:01:21.23
16	2^{-4}	0.70	70.37	363	0:00:20.89	0:01:38.48	0:01:59.37
32	2^{-1}	0.70	71.43	425	0:00:25.80	0:02:15.13	0:02:40.93
64	2^{5}	0.72	72.02	510	0:00:51.69	0:03:36.31	0:04:28.00
128	2^{-1}	0.75	76.82	668	0:01:24.78	0:06:01.50	0:07:26.28
256	2^{-10}	0.73	77.38	98	0:02:22.66	0:10:51.64	0:13:14.30
512	2^{-10}	0.76	80.28	64	0:04:00.63	0:20:20.61	0:24:21.24

Table 9: Pets dataset with SIFT ($l2$-loss SVM): Classification model evaluated on test data for different BoVW dictionary sizes.

Looking at tables Table 8 and Table 9, we observe the F1 score and sensitivity are similar across the $l1$-loss and $l2$-loss SVM. The classifiers for BoVW size 64 and above are considered good classifiers. Although the classifier related to BoVW the size 32 has a good scores, the enormous number of iterations might cause a numerical error. Same as with the Pets Faces dataset, the BoVW size 512 has the best performance scores. This time, for $l2$-loss SVM, the classification model performance scores are slightly better than for the $l1$-loss SVM.

For the Pets dataset, the performance scores of the SIFT features classification show a significant improvement, compared to the the illumination based classification in Section 7.3. For the illumination based classification, the F1 score was 0.57 and the Sensitivity was 56.03%.

However, using the SIFT features we are able to reach F1 score of 0.76 and sensitivity of 80.28%. The confusion matrices for $l1$-loss SVM and $l2$-loss SVM can be seen in Figure 12 an Figure 13, respectively. The reduction in data dimension due to the SIFT feature extraction improved the total classification time significantly.

From these experiments, we can conclude, the SIFT extraction is useful for image classification as it can improve the performance score of an SVM greatly and reduce the total time of training the classifier.

8 Conclusion

In the thesis, we introduced classification of images exploiting feature extraction technique SIFT. The classification performance was compared with a classification performance on raw image data. For classifying both the SIFT features and the raw image data we used an SVM classifier. We faced few challenges of adapting the SIFT features to the classifier.

Firstly, the SIFT feature descriptors describing similar image features had to be grouped together. This was achieved using a k-means clustering algorithm. However, using the algorithm had its own obstacle. We had to analyse different numbers of clusters to decide the size of the BoVW dictionary. The analysis could not be done with convectional methods, both Calinski-Harabaz and Davies-Bouldin indexes provided contradictory results. Therefore, we had decide to select the number experimentally by classifying for various numbers of clusters and choosing the best results.

Next, the images were reshaped to a single vector. This was accomplished using a BoVW approach. Generating a BoVW of the clustered features for each image, the classification could finally be performed.

Based on the classification results, the classification using SIFT features presents potential improvement in score over the classification using the raw image data. Moreover, by reducing the dimension of the classified data, the SIFT feature extraction improves the time needed for training the SVM classifier.

In the future, experimentation with different datasets could be performed.

References

[1] Wikipedia. Computer vision — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Computer%20vision&oldid=894207185. [Online; accessed 28-April-2019].

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] David G. Lowe. Distinctive image featuresfrom scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[4] Mike Stephens Chris Harris. A combined corner and edge detector. *Plessey Research Roke Manor*, 1988.

[5] John F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986.

[6] JustinWick. Comparison of difference of gaussians and mexican hat function, intended to illustrate that dog is a reasonable approximation to mhf. https://commons.wikimedia.org/wiki/File:DOG_vs_MHF.png. [Online; accessed 28-April-2019].

[7] Utkarsh Sinha. Sift: Theory and practice. http://aishack.in/tutorials/sift-scale-invariant-feature-transform-features/. [Online; accessed 28-April-2019].

[8] Wikipedia. k-means clustering. https://en.wikipedia.org/wiki/K-means_clustering. [Online; accessed 28-April-2019].

[9] Sergei Vassilvitskii David Arthur. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[10] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[11] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[12] Wikipedia. List of linguistic example sentences — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=List%20of%20linguistic%20example%20sentences&oldid=891272456. [Online; accessed 28-April-2019].

[13] Vladimir Vapnik Corinna Cortes. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[14] Jakub Kružík, Marek Pecha, Václav Hapla, David Horák, and Martin Čermák. Investigating convergence of linear svm implemented in permonsvm employing mprgp algorithm. In Tomáš Kozubek, Martin Čermák, Petr Tichý, Radim Blaheta, Jakub Šístek, Dalibor Lukáš, and Jiří Jaroš, editors, *High Performance Computing in Science and Engineering*, pages 115–129, Cham, 2018. Springer International Publishing.

[15] David Horák Marek Pecha. Analyzing l1-loss and l2-loss support vector machines implemented in PERMON toolbox. In *Bioanalytical Reviews*, pages 13–23. Springer Berlin Heidelberg, apr 2018.

[16] Fabian Flöck. K-fold cross validation. https://en.wikipedia.org/wiki/Cross-validation_(statistics)#/media/File:K-fold_cross_validation_EN.jpg. [Online; accessed 28-April-2019].

[17] Gary Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[18] Travis Oliphant. NumPy: A guide to NumPy. USA: Trelgol Publishing, 2006–. [Online; accessed 28-April-2019].

[19] Alexandre Gramfort Vincent Michel Bertrand Thirion Olivier Grisel Mathieu Blondel Peter Prettenhofer Ron Weiss Vincent Dubourg Jake Vanderplas Alexandre Passos David Cournapeau Matthieu Brucher Matthieu Perrot Edouard Duchesnay Fabian Pedregosa, Gael Varoquaux. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[20] Marek Pecha Václav Hapla, David Horák. Permonsvm. http://permon.vsb.cz/permonsvm.htm, 2017.

[21] Václav Hapla, David Horák, Martin Čermák, Jakub Kružík, Lukáš Pospíšil, and Radim Sojka. Permonqp. http://permon.it4i.cz/qp/, 2015.

[22] B. F. Smith et al. PETSc users manual. Technical Report ANL-95/11 - Revision 3.5, Argonne National Laboratory, 2016.

[23] Andrew Zisserman C. V. Jawahar Omkar M. Parkhi, Andrea Vedaldi. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[24] IT4Innovations: National Supercomputing Center, VSB-Technical University of Ostrava, Salomon cluster documentation – hardware overview, 2017.

[25] Zdeněk Dostál. *Optimal Quadratic Programming Algorithms, with Applications to Variational Inequalities*, volume 23. SOIA, Springer, New York, US, 2009.

A Confusion Matrices Pets Faces

Confusion matrices for the Pets Faces dataset can be seen in Table 10 for the $l1$-loss SVM and Table 11 for the $l2$-loss SVM.

BoVW Size	TP	FP	FN	TN
2	189	38	190	51
4	183	44	152	89
8	163	64	132	109
16	169	58	105	136
32	167	60	80	161
64	181	46	61	180
128	178	49	71	170
256	161	66	42	199
512	177	50	29	212

Table 10: Pets Faces dataset with SIFT ($l1$-loss SVM): Confusion matrices for different BoVW dictionary sizes.

BoVW Size	TP	FP	FN	TN
2	153	74	154	87
4	162	65	123	118
8	160	67	124	117
16	161	66	98	143
32	166	61	81	160
64	179	48	57	184
128	165	62	57	184
256	166	61	45	196
512	176	51	35	206

Table 11: Pets Faces dataset with SIFT ($l2$-loss SVM): Confusion matrices for different BoVW dictionary sizes.

B Confusion Matrices Pets

Confusion matrices for the Pets Faces dataset can be seen in Table 12 for the $l1$-loss SVM and Table 13 for the $l2$-loss SVM.

BoVW Size	TP	FP	FN	TN
2	154	91	98	148
4	172	73	111	135
8	164	81	104	142
16	164	81	69	177
32	170	75	64	182
64	179	66	68	178
128	179	66	58	188
256	170	75	48	198
512	170	75	44	202

Table 12: Pets dataset with SIFT ($l1$-loss SVM): Confusion matrices for different BoVW dictionary sizes.

BoVW Size	TP	FP	FN	TN
2	142	103	86	160
4	159	86	102	144
8	150	95	97	149
16	171	74	72	174
32	170	75	68	178
64	175	70	68	178
128	179	66	54	192
256	171	74	50	196
512	175	70	43	203

Table 13: Pets dataset with SIFT ($l2$-loss SVM): Confusion matrices for different BoVW dictionary sizes.